1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]

1. It is the computer practice which manages the DS of the linked list containing the first element containing a part for data division, and a pointer part. The phase correct the DS of a linked list, The phase which updates the pointer part of a first element including atomic actuation so that correction of the DS of a linked list may reflect How to contain the phase perform the asynchronous traverse of the DS of a linked list to correction and the coincidence of the DS of a linked list.
2. Computer practice including phase of performing simultaneously two or more asynchronous traverses of DS of linked list according to claim 1.
3. Computer practice including phase where atomic actuation corrects pointer part of first element, and points out the 2nd element including phase where phase of correcting DS of linked list adds the 2nd element to DS of linked list according to claim 1.
4. Computer practice including phase where atomic corrective action corrects pointer part of first element, and points out the 3rd element according to claim 1.
5. Computer practice including phase where atomic actuation corrects pointer part of first element, and points out the 3rd element instead of the 2nd element including phase where phase of correcting DS of linked list removes the 2nd element from DS of linked list according to claim 1.
6. Computer practice including phase of performing mark / sweep garbage collection actuation to DS of linked list to correction and updating actuation, and coincidence according to claim 1.
7. Computer practice according to claim 1 whose phase of correcting DS of linked list is synchronous operation.
8. Computer practice according to claim 1 in which asynchronous traverse includes retrieval actuation which searches part for data division of element within DS of linked list.
9. In Object of Linked List in Object Oriented Programming Environment DS of Linked List Containing Two or More Elements Which Have a Part for Data Division, and Pointer Part, Respectively, The correction method which corrects the DS of a linked list by insertion or clearance of first element, The updating method reflecting correction of the DS of the linked list which updates the pointer part of the 2nd element, and uses an atomic corrective action, the object containing the traverse method which performs correction of the DS of the linked list by correction method, simultaneously the asynchronous traverse of the DS of a linked list.
10. It is the Processor when Processor Performs. By Inserting or Removing 2nd Element The phase of correcting the DS of the linked list containing the first element which has a part for data division, and a pointer part, The phase which updates the pointer part of a first element including atomic actuation so that correction of the DS of a linked list may be reflected, The computer-readable medium which memorized a series of instructions which perform correction of the DS of the list linked [ ], and the phase of performing the asynchronous traverse of the DS of a linked list to coincidence.
11. The computer-readable medium according to claim 10 which memorized correction of the DS of a linked list, and a series of instructions which make coincidence perform two or more asynchronous traverses of the DS of a linked list to a processor.
12. The computer-readable medium which is a computer-readable medium according to claim 10 which memorizes a series of instructions to which the 2nd element is made to add to a processor into the DS of a linked list, and includes the phase where atomic actuation corrects the pointer

part of a first element, and points out the 2nd element.
13. A computer-readable medium including the phase where atomic actuation corrects the pointer part of a first element, and points out the 3rd element according to claim 10.
14. The computer-readable medium by which it is the computer-readable medium according to claim 10 which memorizes a series of instructions from which a processor is made to remove the 2nd element from the DS of a linked list, and the phase to update includes the phase which corrects the pointer part of a first element to ATOMIKKU, and points out the 3rd element instead of a first element.
15. A computer-readable medium including the phase of performing mark / sweep garbage collection actuation to the DS of a linked list to correction and updating actuation, and coincidence according to claim 10.
16. The computer-readable medium according to claim 10 whose phase of correcting the DS of a linked list is synchronous operation.
17. In order to manage the DS of the linked list containing the element containing a part for data division, and a pointer part The phase perform the 1st actuation to the DS of a linked list, The approach, by which it is a computer practice including the 1st actuation and the phase perform the asynchronous traverse of the DS of a linked list to coincidence, and this read is performed as atomic actuation including the phase which this asynchronous traverse performs in the read of the pointer part of an element.,
18. The computer practice according to claim 17 whose 1st actuation is the asynchronous traverse of the DS of a linked list further.
19. The computer practice according to claim 17 whose 1st actuation is synchronous correction of the DS of a linked list further.
20. It is the Processor when Processor Performs. Phase of Performing 1st Actuation to DS of Linked List, It is the computer-readable medium which memorized a series of instructions which perform the 1st actuation, and the phase of performing the asynchronous traverse of the DS of a linked list to coincidence. The computer-readable medium by which this read is performed as ATOMITTA actuation including the phase where this asynchronous traverse performs read of the pointer part of an element.
21. The computer-readable medium according to claim 20 whose 1st actuation is the asynchronous traverse of the DS of a linked list.
22. The computer-readable medium according to claim 20 whose 1st actuation is synchronous correction of the DS of a linked list further.
23. It is the Processor when Processor Performs. By Inserting or Removing 2nd Element The phase of correcting the DS of the linked list containing the first element which has a part for data division, and a pointer part, The phase which updates the pointer part of a first element including atomic actuation so that correction of the DS of a linked list may be reflected, The computer data signal showing a series of instructions which perform correction of the DS of the list linked [ ], and the phase of performing the asynchronous traverse of the DS of a linked list to coincidence built into the subcarrier.
24. It is the Processor when Processor Performs. Phase of Performing 1st Actuation to DS of Linked List Containing Element Containing a Part for Data Division, and Pointer Part, Express a series of instructions which perform the 1st actuation, and the phase of performing the asynchronous traverse of the DS of a linked list to coincidence. The computer data signal with which it is the computer data signal built into the subcarrier, and this read is performed as atomic actuation including the phase where this asynchronous traverse performs read of the pointer part of an element.

[Translation done.]

DETAILED DESCRIPTION

[Detailed Description of the Invention]
The method of managing the DS of a linked list, and field of equipment invention Generally, this invention relates to the field of a computer software program, and relates to the approach of managing the DS of a linked list maintained and accessed by the detail by the computer program.
Background The DS of a linked list can store the sequence of a data element, can add a data element to these sequences promptly, and can remove it from them. DS of a linked list is characterized by each of that element containing the pointer to a series of next elements under linked list.
Therefore, when the element to the DS of a linked list is added or an element is removed from there, he updates the pointer of the last element and it will be understood that correction of the DS of a linked list must be made to reflect.
A multithread program may need access to the DS of the linked list of specification [some of them] including some threads. Such a thread performs non-corrective actions, such as GET, or corrective actions, such as ADD, to the DS of a linked list. In order to prevent the concurrent access from some threads to an object like the DS of a linked list, by such thread, lock treatment is performed to a target object, and thereby, generally, operation of the so-called "synchronous" method which prevents that the method of arbitration accesses a target object is performed until the method in activity carries out lock discharge of the object again.
When a target object is the DS of a linked list, this whole DS is locked by that method in this way until a synchronous method is usually completed. Therefore, the concurrent access from various threads to the DS of a linked list is forbidden.
Outline of invention According to the 1st mode of this invention, the method of managing the DS of a linked list is offered. The DS of a linked list contains the first element which has a part for data division, and a pointer part both. This approach needs to correct the DS of a linked list by inserting or removing the 2nd element. The pointer part of a first element is updated and reflects correction of the DS of a linked list. The renewal of the pointer part of this first element includes atomic actuation. correction of the DS of a linked list, simultaneously the asynchronous traverse of the DS of a linked list are performed. In the one embodiment, traverse actuation is a non-corrective action and includes the read of data, and/or retrieval actuation.
According to the 2nd mode of this invention, when a processor performs, the computer-readable medium which memorized a series of instructions which make the processor perform each phase of an above-mentioned approach is offered.
According to the 3rd mode of this invention, a linked list object is offered in an object oriented programming environment. This object contains the correction method which corrects the DS of a linked list by insertion or clearance of the DS of a linked list, and a first element. This object also contains the updating method reflecting the correction which updated the pointer part of the 2nd element and was carried out to the DS of a linked list by the correction method. The updating phase performed by this updating method includes atomic actuation. This object also contains in the correction and coincidence of the DS of a linked list by the correction method the traverse method which performs the asynchronous traverse of the DS of a linked list. This traverse can perform inspection of the pointer part of a first element as atomic actuation.
The description of others of this invention will become clear from the drawing of the following attachment, detailed explanation, and a claim.

Easy explanation of a drawing This invention is illustrated to each drawing of an attached drawing not for the purpose of a limit but for the purpose of instantiation. By a diagram, the same reference mark points out some elements.
Drawing 1 is a block diagram showing the multithread program which operates in an object-oriented program environment.
Drawing 2 is a schematic diagram showing the DS of a linked list.
the -- the [3a drawing and] -- 3b drawing is a schematic diagram showing the element insertion actuation and element clearance actuation which are performed to the DS of the linked list shown in drawing 2, respectively.
Drawing 4 is a flow chart showing how to manage the DS of the linked list by 1 operation gestalt of this invention.
Drawing 5 is a block diagram showing the structure of a hash table.
Drawing 6 is a block diagram showing a series of hash table objects built according to 1 operation gestalt of this invention.
Drawing 7 is a block diagram showing the computer system containing the computer-readable medium which memorized a series of instructions which make the processor perform the approach of managing the DS of the linked list by 1 operation gestalt of this invention, when the processor of computer system performs.
Detailed explanation The computer practice which manages the DS of a linked list is described. In the following description, it aims at explanation, he indicates the specific detail of shoes, and this invention is understood thoroughly. However, probably, it will be clear to this contractor that this invention can be carried out without using these details. Although indicating below one instantiation operation gestalt carried out in object oriented programming language, speaking concretely, this invention is not this language or the thing limited to the programming language type of arbitration in addition to this.
When drawing 1 is referred to, they are U.S. California and Mountain. Sun of View The block diagram of the multithread program (10) created with object-oriented programming language, such as JavaTM programming language which Microsystems developed, is shown. A program 10 maintains the thread queue 12 which needs access to an object 16 and which consists of threads 14b, 14c, and 14d. Thread 14a is shown as what has access to current and an object 16. The multithread program 10 is characterized by the ability to access to the common data with which each thread 14 shares data and may be contained in objects, such as an object 16, by that cause, or a variable. For example, in JavaTM programming language, a Java virtual machine (JVM) can always support activation of many threads. These threads perform the Java code which acts on the value and object which reside permanently into a shared memory independently, respectively. Two or more threads can be supported by having two or more hardware processings, carrying out the time slicing of the single hardware processor, or carrying out the time slicing of many hardware processors.
It can access the thread 14 of an active state to the object of the arbitration which has reference. For example, thread 14 a-d has the reference to an object 16, respectively. When two or more threads access to a common object, the actuation performed by each of these threads collides, and is considered that it may lead to malfunction of the thread of destruction of a related object, one side, or both. In order to cope with this problem, generally specifying "synchronous" access as the object which two or more threads share is performed. Speaking concretely, the vocabulary "a synchronization" pointing out the capacity it enables it to complete certainly, without one thread having the actuation to an object, a variable, or data interrupted. It can synchronize by using a semaphore or mutexes. As an exception method, the synchronization between thread accesses can also be obtained by using the "monitor" which is the high level device in which only only one thread enables it to perform at once the coding region protected by the related monitor. In JavaTM language, the object which can be again notified to the thread which blocked the thread, and was blocked when it became available after that is called a monitor object. For example, if drawing 1 is referred to, since an object 16 will turn into a monitor object, in that case, when 14a accesses to an object 16, thread 14 b-c is blocked, and when thread 14a completes access to an object 16, it will notify to the thread queue 12. The treatment which the treatment which blocks access to the object by other threads will be called if a "lock" is performed, and enables access to the object by

another thread again is called "lock discharge" actuation.
It will be understood that there is disadvantage of the engine performance in which only a single thread can always access an object, therefore specific parallel operation may be forbidden to the thread which performs a synchronous method.
The common data structure type incorporated in the object 16 is DS 20 of the linked list shown in drawing 2. DS 20 of this linked list makes easy quick insertion of the element to this sequence, and quick clearance of the element from there including an elements [which were able to be set in order / 22a–22n] sequence. Each element 22 contains the pointer 26 to the following sequential element 22 in a data item 24 and DS 20 of a linked list. Element 22n of the last in DS 20, it has a nullpointer showing termination of a list.
the -- the [3a drawing and] -- 3b drawing is drawing showing the modification voice of DS 20 of a linked list. if it says concretely -- the -- 3a drawing is drawing showing insertion of the element 22 in DS 20 (n), and the element 22 (n+1) of a between [22 (n+2)]. the -- 3b drawing is drawing showing clearance of the element 22 (n+1) from DS 20 of a linked list similarly. first -- the -- if 3a drawing is referred to, the thread of an active state which has access to an object including DS 20 of a linked list can insert an element 22 (n+1). In order to insert such an element in DS 20 of a linked list (or addition), it is necessary to update so that the element 22 (n+1) in which the pointer of the element 22 (n) which is just before the location which is going to insert an element 22 (n+1) was newly inserted from the former sequential element 22 (n+2) may be pointed out. An arrow head 28 shows renewal of the pointer of this element 22 (n). The inserted pointer of an element 22 (n+1) must also be set so that an element 22 (n+2) may be pointed out.
With 1 operation gestalt of this invention, each pointer 26 includes 32 bit addresses. The need for synchronous in a program will become clear if assumption that two threads have accessed DS 20 of a linked list simultaneously is taken into consideration. Speaking concretely, the 1st thread which is inserting the element 22 (n+1) updating the pointer 26 of an element 22 (n), and taking into consideration the case where he is trying to point out the element 22 (n+1) inserted newly. The traverse of DS 20 may be performed by inspecting the data pass with which it could come, simultaneously the 2nd thread was specified by it. Speaking concretely, the 2nd thread's having traversed DS 20, in order to search the predetermined data item 24. What is understood are [the read actuation performed on the pointer corrected selectively] disadvantageous for the traverse of DS 20 by the 2nd thread will be understood since the 2nd thread turns it to the next element in a sequence depending on the pointer of each element. In order to prevent the scenario outlined above, the method of the object which accesses DS 20 of a linked list is synchronized, and, generally making it only only one thread or a method always have access to DS 20 is performed. Speaking concretely, being able to attain a lock directly using a synchronization on the DS of the hash table which refers to the DS of a linked list, or DS 20 of a linked list. However, it will be understood at once only by [one] carrying out a method and preventing from accessing the DS of a loan NKU finishing list that optimum performance is no longer offered. the meaning of this invention is understood -- being alike -- it must take into consideration about three following actuation which can be performed to DS 20 of a linked list. namely, -- the [1.] -- as shown in 3a drawing, an element is added to DS 20 of a linked list -- insertion (or addition) actuation the [2.] -- as shown in 3b drawing, an element is removed from DS 20 -- clearance (or deletion) actuation -- and -- 3. -- traverse actuation in which it decides that a certain specific data item exists in it, and the content of the element within DS 20 is inspected in order to search such a data item depending on/or the case.
Traverse actuation is distinguished from insertion actuation and clearance actuation in that the correction to DS 20 does not take place. This invention proposes the approach that the non-correcting traverse of DS 20 of a linked list is as asynchronous as other traverse actuation and corrective actions. If the asynchronous traverse actuation is carried out, since a corrective action and traverse actuation can take place simultaneously and some asynchronous traverses of DS 20 of a linked list can also take place to coincidence, it is advantageous. Therefore, in the program two or more threads of whose mainly perform traverse actuation about DS 20 of a linked list, the thread of these large number can access DS 20 simultaneously. Implementation of asynchronous traverse actuation becomes easy by specifying the ATOMIKKU read and the write-in actuation by the thread of the pointer 26 of an element 22. Speaking concretely, completing atomic actuation

certainly indivisible therefore functionally. next — the methodology of this invention — the — the [3a drawing and] — it describes especially with reference to 3b drawing. the — the insertion method shown in 3a drawing synchronizes with other synchronous methods, and includes the following phase especially.

1. Phase of inspecting whether element which it is going to insert already existing in the linked list structure 20. It closes insertion actuation, in existing.

2. Phase which updates pointer of element 22 (n+1) inserted in atomic actuation, and points out following sequential element 22 (n+2). If this phase is completed, both the element 22 (n) and 22 (n+1) have pointed out the same element 22 (n+2).

3. Phase which updates pointer of element 22 (n) and points out inserted element 22 (n+1) in atomic actuation.

Similarly, the synchronous method which removes an element 22 from DS 20 of a linked list can include the following phase especially.

1. Phase of deciding whether inspection actuation being performed and related data item being actually included during linked list.

2. Phase which sets pointer of element 22 (n) in atomic actuation so that element 22 (n+2) may be pointed out.

Please care about that correction of the pointer 26 of an element 22 includes atomic actuation in the actuation performed above. Therefore, inspecting data pass with any effective methods which perform traverse actuation of DS 20 of a linked list between such insertion or deletion actuation will be guaranteed. The guarantee of this effective-data pass makes easy access to DS 20 of the linked list by the method which carries out asynchronous traverse actuation. However, with 1 operation gestalt, it will be understood that a synchronous corrective action is still required in order to prevent the collision between such corrective actions. Therefore, some threads incorporating such an asynchronous traverse method are enabled to access DS 20 of a single linked list simultaneously by using an asynchronous traverse method, and it becomes possible to optimize the engine performance of a computer program in this way.

Furthermore, this invention is so-called "mark / sweep garbage collection."

It is suitable for especially the activity with the programming language which has the capacity to carry out. Speaking concretely, with such a mark / a sweep garbage collection technique, not mounting a reference counter style, in order to maintain the tracking of the reference to a specific object. The garbage collection technique using a reference count may restrict the advantage on the engine performance which this invention attains, when synchronous operation is needed, therefore synchronous garbage collection actuation is performed.

Next, reference of drawing 4 shows the method 30 of managing DS 20 of the linked list containing some elements 22 by this invention. It starts from a phase 32, and this approach advances and performs 2 sets of actuation to juxtaposition. Speaking concretely, by the approach 30, the 1st thread's being able to correct DS 20 of a linked list by inserting or removing an element 22 in a phase 34. Subsequently, an approach 30 goes to a phase 36, corrects to ATOMIKKU the pointer of an element 22 with which it is related within DS 20, and reflects the correction made in the phase 34. With 1 operation gestalt of this invention, the programming language which makes easy ATOMIKKU correction and read of a pointer 26 is JavaTM programming language. By the approach 30 of this invention, asynchronous traverse actuation of the number of arbitration of DS 20 of a linked list can be performed in parallel to performing phases 34 and 36 and coincidence in a phase 38.

Drawing 5 is drawing containing some the entries or buckets 42 which can refer to DS 20 of each linked list, respectively showing a hash table 40. According to hash table DS, the data item (for example, employee record) used as an object is identified by the key. For example, the data item in the first element 22 of DS 20a of a linked list is identified by the key "AAA." A hash table 40 calculates the integral value which is called a hash code and which is obtained from a key. Therefore, the hash code generated from the key "AAA" of the example of a graphic display serves as zero (0). In this way, a hash table 40 makes generation of the shortened array index easy. It is thought that the close data item which has the key by which the common hash code is generated is in the same bucket, and DS 20 of the linked list of what is shown in drawing 5 can be included. This invention can be used in order to manage DS 20 of the linked list referred to with a

hash table 40.
Drawing 6 is drawing by this invention showing a series of objects 50 containing the object of a class 52. Each object 50 is shown as a thing including the hash table structure 40 which attaches an index to DS 20 of some linked lists. Each object contains further three asynchronous traverse methods 60, i.e., the "CONTAINS" method, the "CONTAINS KEY" method 62, and the "GET" method 64. The "CONTAINS" method 60 tests whether the map of the key given by the thread is carried out to the value as which it was specified in hash table DS 40. The "CONTAINS KEY" method 62 tests whether the specified object is a key in hash table DS 40, and the "GET" method 64 returns the data item to which the map of the given key is carried out in hash table DS 40. Each object contains further the synchronous "PUT" method 66 which can insert an element 22 into DS 20 of a linked list according to above-mentioned methodology. Each object 50 contains further the synchronous "REMOVE" method 68 which can remove an element 22 from DS 20 of a linked list according to above-mentioned methodology. Of course, the above-mentioned object 50 is mere instantiation, and it will be understood that the principle of this invention is applicable to the object of the number of arbitration of various structures.
Next, the computer system 70 containing a processor 72, static memory 74, and main memory 76 is shown in drawing 7. A processor 72 and the memory 74 and 76 of each other [and] are connected with some peripheral devices through a bus 78. The actuation unit 86 which holds the video displays 80 (a cathode-ray tube (CTR), liquid crystal display (LCD), etc.), the alphabetic-character input devices 82 (keyboard etc.), cursor controllers 84 (mouse etc.), and the computer-readable medium 88, the signal generation equipments 90 (one pair of voice loudspeakers etc.), and network interface equipment 94 are included in an above-mentioned peripheral device. In this invention, the vocabulary a "computer-readable medium" shall point out the magnetic storage which can obtain data for the actuation unit 86, main memory 76, static memory 74, a processor 72, or a processor 72 to perform and which can be held into the medium of arbitration in addition to this. The hash table object 92 containing methods, such as an above-mentioned thing, is shown as what resides permanently completely selectively into the computer-readable medium 88, main memory 76, or processor 72 the very thing. The hash table object 92 includes a series of instructions which make a processor 72 perform the phase described in relation to drawing 4 at least, when a processor 72 performs.
Network interface equipment 94 can use a modem, a network adapter card, or computer system 70 as the equipment of the other arbitration combined with a computer network. Network access equipment 94 can be used, and the subcarrier by which the computer data signal was coded can be generated or received. A computer data signal can be interpreted and the program code which can be performed in order to carry out this invention can be generated.
Thus, how to manage the DS of a linked list was described. Although this invention was explained in relation to the specific instantiation operation gestalt, probably, it will be clear that various corrections and modification can be added to these operation gestalten, without deviating from the extensive main point and extensive range of this invention. Therefore, please this description and a drawing be not restrictive and consider that they are an instantiation-thing.

[Translation done.]

(19)日本国特許庁（ＪＰ）　(12) **公表特許公報（Ａ）**　(11)特許出願公表番号
特表2002－501649
（P2002－501649A）
(43)公表日　平成14年1月15日(2002.1.15)

(51)Int.Cl.[7]		識別記号	FI		テーマコード(参考)
G06F	12/00	590	G06F	12/00	590
	9/44	530		9/44	530S
	9/46	340		9/46	340C

審査請求　未請求　予備審査請求　有　（全 25 頁）

(21)出願番号　特願平11－509842
(86)(22)出願日　平成10年6月30日(1998.6.30)
(85)翻訳文提出日　平成11年3月1日(1999.3.1)
(86)国際出願番号　ＰＣＴ／ＵＳ９８／１３６５２
(87)国際公開番号　ＷＯ９９／００７２５
(87)国際公開日　平成11年1月7日(1999.1.7)
(31)優先権主張番号　０８／８８７，３３９
(32)優先日　平成９年６月30日(1997.6.30)
(33)優先権主張国　米国（ＵＳ）
(81)指定国　ＥＰ(ＡＴ，ＢＥ，ＣＨ，ＣＹ，ＤＥ，ＤＫ，ＥＳ，ＦＩ，ＦＲ，ＧＢ，ＧＲ，ＩＥ，ＩＴ，ＬＵ，ＭＣ，ＮＬ，ＰＴ，ＳＥ)，ＣＮ，ＪＰ，ＫＲ

(71)出願人　サン・マイクロシステムズ・インコーポレーテッド
アメリカ合衆国・94303・カリフォルニア州・パロ　アルト・サン　アントニオ　ロード・901
(72)発明者　クラッジ，ケヴィン・エル
アメリカ合衆国・95014・カリフォルニア州・カッパチーノ・バークウッド　ドライブ・10190
(74)代理人　弁理士　山川　政樹　（外５名）

(54)【発明の名称】　リンク済みリストのデータ構造を管理する方法および装置

(57)【要約】
リンク済みリストのデータ構造（２０）を管理する方法が開示されている。このリンク済みリストのデータ構造（２０）は、データ項目（２４ａ、２４ｂ…２４ｎ）および順次続くエレメントへのポインタ（２６ａ、２６ｂ…２６ｎ）それぞれ合むいくつかのエレメント（２２ａ、２２ｂ…２２ｎ）を有する。この方法では、リンク済みリストのデータ構造（２０）に対する同時かつ非同期トラバース動作を可能にしながら、エレメントをそこに挿入する、またはそこから除去することによってリンク済みリストのデータ構造（２０）を修正する（３４）ことが可能となる。具体的に言うと、この方法では、リンク済みリストのデータ構造（２０）内のエレメント（２４ａ、２４ｂ…２４ｎ）のポインタ（２６ａ、２６ｂ…２６ｎ）をアトミック動作を使用して修正し（３６）、リンク済みリストのデータ構造（２０）に行われた任意の修正を反映する必要がある。アトミック動作を利用してポインタ（２６ａ、２６ｂ…２６ｎ）を更新することで、非同期トラバース動作（３８）が有効データ・パスを検査することが保証される。

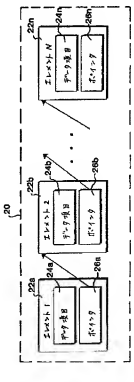

FIG. 2

【特許請求の範囲】

1．データ部分およびポインタ部分を含む第１エレメントを含むリンク済みリストのデータ構造を管理するコンピュータ実施方法であって、

リンク済みリストのデータ構造を修正する段階と、

アトミック動作を含み、リンク済みリストのデータ構造の修正を反映するように第１エレメントのポインタ部分を更新する段階と

リンク済みリストのデータ構造の修正と同時にリンク済みリストのデータ構造の非同期トラバースを実行する段階と

を含む方法。

2．リンク済みリストのデータ構造の複数の非同期トラバースを同時に実行する段階を含む請求項１に記載のコンピュータ実施方法。

3．リンク済みリストのデータ構造を修正する段階がリンク済みリストのデータ構造に第２エレメントを追加する段階を含み、アトミック動作が、第１エレメントのポインタ部分を修正して第２エレメントを指すようにする段階を含む請求項１に記載のコンピュータ実施方法。

4．アトミック修正動作が、第１エレメントのポインタ部分を修正して第３エレメントを指すようにする段階を含む請求項１に記載のコンピュータ実施方法。

5．リンク済みリストのデータ構造を修正する段階がリンク済みリストのデータ構造から第２エレメントを除去する段階を含み、アトミック動作が、第１エレメントのポインタ部分を修正して、第２エレメントの代わりに第３エレメントを指すようにする段階を含む請求項１に記載のコンピュータ実施方法。

6．修正および更新動作と同時に、リンク済みリストのデータ構造に対してマーク／スイープ・ガーベージ・コレクション動作を実行する段階を含む請求項１に記載のコンピュータ実施方法。

7．リンク済みリストのデータ構造を修正する段階が同期動作である請求項１に記載のコンピュータ実施方法。

8．非同期トラバースが、リンク済みリストのデータ構造内のエレメントのデータ部分を検索する検索動作を含む請求項１に記載のコンピュータ実施方法。

９．オブジェクト指向プログラミング環境におけるリンク済みリストのオブジェクトにおいて

データ部分およびポインタ部分をそれぞれ有する複数のエレメントを含むリンク済みリストのデータ構造と、

第１エレメントの挿入または除去によってリンク済みリストのデータ構造を修正する修正メソッドと、

第２エレメントのポインタ部分を更新して、アトミック修正動作を利用するリンク済みリストのデータ構造の修正を反映する更新メソッドと、

修正メソッドによるリンク済みリストのデータ構造の修正と同時に、リンク済みリストのデータ構造の非同期トラバースを実行するトラバース・メソッドとを含むオブジェクト。

１０．プロセッサによって実行されたときに、そのプロセッサに、

第２エレメントを挿入または除去することによって、データ部分およびポインタ部分を有する第１エレメントを含むリンク済みリストのデータ構造を修正する段階と、

アトミック動作を含み、リンク済みリストのデータ構造の修正を反映するように第１エレメントのポインタ部分を更新する段階と、

リンク済みリストのデータ構造の修正と同時にリンク済みリストのデータ構造の非同期トラバースを実行する段階と

を実行させる一連の命令を記憶したコンピュータ可読媒体。

１１．リンク済みリストのデータ構造の修正と同時にリンク済みリストのデータ構造の複数の非同期トラバースをプロセッサに実行させる一連の命令を記憶した請求項１０に記載のコンピュータ可読媒体。

１２．プロセッサに第２エレメントをリンク済みリストのデータ構造中に追加させる一連の命令を記憶する請求項１０に記載のコンピュータ可読媒体であって、アトミック動作が第１エレメントのポインタ部分を修正して第２エレメントを指すようにする段階を含むコンピュータ可読媒体。

１３．アトミック動作が、第１エレメントのポインタ部分を修正して第３エレメントを指すようにする段階を含む請求項１０に記載のコンピュータ可読媒体。

１４．プロセッサに第２エレメントをリンク済みリストのデータ構造から除去させる一連の命令を記憶する請求項１０に記載のコンピュータ可読媒体であって、更新する段階が、第１エレメントのポインタ部分をアトミックに修正して、第１エレメントの代わりに第３エレメントを指すようにする段階を含むコンピュータ可読媒体。

１５．修正および更新動作と同時に、リンク済みリストのデータ構造に対してマーク／スイープ・ガーベージ・コレクション動作を実行する段階を含む請求項１０に記載のコンピュータ可読媒体。

１６．リンク済みリストのデータ構造を修正する段階が同期動作である請求項１０に記載のコンピュータ可読媒体。

１７．データ部分およびポインタ部分を含むエレメントを含むリンク済みリストのデータ構造を管理するための、

リンク済みリストのデータ構造に対して第１動作を実行する段階と、

第１動作と同時にリンク済みリストのデータ構造の非同期トラバースを実行する段階とを含むコンピュータ実施方法であって、この非同期トラバースがエレメントのポインタ部分の読取りを実行する段階を含み、この読取りがアトミック動作として実行される方法。

１８．第１動作がさらにリンク済みリストのデータ構造の非同期トラバースである請求項１７に記載のコンピュータ実施方法。

１９．第１動作がさらにリンク済みリストのデータ構造の同期修正である請求項１７に記載のコンピュータ実施方法。

２０．プロセッサによって実行されたときに、そのプロセッサに、

リンク済みリストのデータ構造に対して第１動作を実行する段階と、

第１動作と同時にリンク済みリストのデータ構造の非同期トラバースを実行する段階と

を実行させる一連の命令を記憶したコンピュータ可読媒体であって、

この非同期トラバースがエレメントのポインタ部分の読取りを実行する段階を含み、この読取りがアトミッタ動作として実行されるコンピュータ可読媒体。

２１．第１動作がリンク済みリストのデータ構造の非同期トラバースである請求

項２０に記載のコンピュータ可読媒体。

２２．第１動作がさらにリンク済みリストのデータ構造の同期修正である請求項２０に記載のコンピュータ可読媒体。

２３．プロセッサによって実行されたときに、そのプロセッサに、

第２エレメントを挿入または除去することによって、データ部分およびポインタ部分を有する第１エレメントを含むリンク済みリストのデータ構造を修正する段階と、

アトミック動作を含み、リンク済みリストのデータ構造の修正を反映するように第１エレメントのポインタ部分を更新する段階と、

リンク済みリストのデータ構造の修正と同時にリンク済みリストのデータ構造の非同期トラバースを実行する段階と

を実行させる一連の命令を表す、搬送波に組み込まれたコンピュータ・データ信号。

２４．プロセッサによって実行されたときに、そのプロセッサに、

データ部分およびポインタ部分を含むエレメントを含むリンク済みリストのデータ構造に対して第１動作を実行する段階と、

第１動作と同時にリンク済みリストのデータ構造の非同期トラバースを実行する段階と

を実行させる一連の命令を表す、搬送波に組み込まれたコンピュータ・データ信号であって、

この非同期トラバースがエレメントのポインタ部分の読取りを実行する段階を含み、この読取りがアトミック動作として実行されるコンピュータ・データ信号。

【発明の詳細な説明】

リンク済みリストのデータ構造を管理する方法および装置

発明の分野

本発明は一般に、コンピュータ・ソフトウェア・プログラムの分野に関し、詳細には、コンピュータ・プログラムによって維持およびアクセスされるリンク済みリストのデータ構造を管理する方法に関する。

背景

リンク済みリストのデータ構造は、データ・エレメントのシーケンスを格納し、データ・エレメントを迅速にこれらのシーケンスに追加し、それらから除去することができる。リンク済みリストのデータ構造は、その各エレメントが、リンク済みリスト中の次の一連のエレメントへのポインタを含むことを特徴とする。したがって、リンク済みリストのデータ構造へのエレメントを追加する、またはそこからエレメントを除去したときには、直前のエレメントのポインタを更新し、リンク済みリストのデータ構造の修正を反映させなければならないことが理解されよう。

マルチスレッド・プログラムはいくつかのスレッドを含み、そのうちのいくつかが特定のリンク済みリストのデータ構造へのアクセスを必要とすることもある。こうしたスレッドは、ＧＥＴなどの非修正動作、またはＡＤＤなどの修正動作をリンク済みリストのデータ構造に対して実行する。いくつかのスレッドからリンク済みリストのデータ構造のようなオブジェクトへの同時アクセスを防止するために、このようなスレッドでは、目標オブジェクトにロック処置を実行し、それにより、活動中のメソッドがそのオブジェクトを再度ロック解除するまで、その他任意のメソッドが目標オブジェクトにアクセスすることを防止する、いわゆる「同期」メソッドの実施が一般に行われている。

目標オブジェクトがリンク済みリストのデータ構造である場合には、このようにこのデータ構造全体は、通常は同期メソッドが完了するまでそのメソッドによってロックされる。したがって、様々なスレッドからリンク済みリストのデータ構造への同時アクセスは禁止される。

発明の概要

本発明の第１の態様によれば、リンク済みリストのデータ構造を管理する方法が提供される。リンク済みリストのデータ構造は、データ部分およびポインタ部分を両方とも有する第１エレメントを含む。この方法は、第２エレメントを挿入または除去することによってリンク済みリストのデータ構造を修正することを必要とする。第１エレメントのポインタ部分は更新され、リンク済みリストのデータ構造の修正を反映する。この第１エレメントのポインタ部分の更新は、アトミック動作を含む。リンク済みリストのデータ構造の修正と同時に、リンク済みリストのデータ構造の非同期トラバースが実行される。１実施態様では、トラバース動作は非修正動作であり、データの読取りおよび／または検索動作を含む。

本発明の第２の態様によれば、プロセッサによって実行されたときにそのプロセッサに上述の方法の各段階を実行させる一連の命令を記憶した、コンピュータ可読媒体が提供される。

本発明の第３の態様によれば、オブジェクト指向プログラミング環境中でリンク済みリストオブジェクトが提供される。このオブジェクトは、リンク済みリストのデータ構造、および第１エレメントの挿入または除去によってリンク済みリストのデータ構造を修正する修正メソッドを含む。このオブジェクトは、第２エレメントのポインタ部分を更新して修正メソッドによってリンク済みリストのデータ構造に行われた修正を反映する更新メソッドも含む。この更新メソッドによって実行される更新段階は、アトミック動作を含む。このオブジェクトは、修正メソッドによるリンク済みリストのデータ構造の修正と同時にリンク済みリストのデータ構造の非同期トラバースを実行するトラバース・メソッドも含む。このトラバースは、アトミック動作として第１エレメントのポインタ部分の検査を実行することができる。

本発明のその他の特徴は、以下の添付の図面、詳細な説明、および請求の範囲から明らかになるであろう。

図面の簡単な説明

本発明を、制限ではなく例示を目的として、添付の図面の各図に図示する。図

では同じ参照符がいくつかの要素を指す。

第１図は、オブジェクト指向プログラム環境で動作するマルチスレッド・プログラムを示すブロック図である。

第２図は、リンク済みリストのデータ構造を示す概略図である。

第３ａ図および第３ｂ図は、第２図に示すリンク済みリストのデータ構造に対して実行されるエレメント挿入動作およびエレメント除去動作をそれぞれ示す概略図である。

第４図は、本発明の１実施形態によるリンク済みリストのデータ構造を管理する方法を示す流れ図である。

第５図は、ハッシュ・テーブルの構造を示すブロック図である。

第６図は、本発明の１実施形態に従って構築された一連のハッシュ・テーブル・オブジェクトを示すブロック図である。

第７図は、コンピュータ・システムのプロセッサによって実行されたときに、本発明の１実施形態によるリンク済みリストのデータ構造を管理する方法をそのプロセッサに実行させる一連の命令を記憶したコンピュータ可読媒体を含むコンピュータ・システムを示すブロック図である。

詳細な説明

リンク済みリストのデータ構造を管理するコンピュータ実施方法について記述する。以下の記述では、説明を目的としていくつかの特定の詳細を記載し、本発明が完全に理解されるようにする。ただし、これらの詳細を用いずに本発明を実施することができることは当業者には明らかであろう。具体的に言うと、オブジェクト指向プログラミング言語で実施される１つの例示的な実施形態を以下に記載するが、本発明は、この言語またはその他任意のプログラミング言語タイプに限定されるものではない。

第１図を参照すると、米国カリフォルニア州、Ｍｏｕｎｔａｉｎ　Ｖｉｅｗの

Ｓｕｎ　Ｍｉｃｒｏｓｙｓｔｅｍｓが開発したＪａｖａ™プログラミング言語などのオブジェクト指向プログラム言語で作成されたマルチスレッド・プログラム（１０）のブロック図が示されている。プログラム１０は、オブジェクト１６へ

のアクセスを必要とする、スレッド１４ｂ、１４ｃ、および１４ｄからなるスレッド待ち行列１２を維持する。スレッド１４ａは現在、オブジェクト１６へのアクセスを有するものとして示されている。マルチスレッド・プログラム１０は、各スレッド１４がデータを共用し、それによりオブジェクト１６などのオブジェクト中に含まれることがある共通のデータや変数へアクセスすることができることを特徴とする。例えば、Ｊａｖａ™ プログラミング言語では、Ｊａｖａ仮想計算機（ＪＶＭ）は、常に多数のスレッドの実行をサポートすることができる。これらのスレッドはそれぞれ独立して、共用メモリ中に常駐する値およびオブジェクトに作用するＪａｖａコードを実行する。複数のスレッドは、複数のハードウェア処理を有するか、単一のハードウェア・プロセッサをタイム・スライシングするか、または多数のハードウェア・プロセッサをタイム・スライシングすることによってサポートすることができる。

活動状態のスレッド１４は、それが参照を有する任意のオブジェクトへアクセスできる。例えば、スレッド１４ａ～ｄはそれぞれ、オブジェクト１６への参照を有する。２つ以上のスレッドが共通のオブジェクトへアクセスする場合には、これらのスレッドそれぞれによって実行される動作が衝突し、関連するオブジェクトの破壊、あるいは一方または両方のスレッドの誤動作につながることがあると考えられる。この問題に対処するために、複数のスレッドが共用するオブジェクトに、「同期」アクセスを指定することが一般に行われている。具体的に言うと、「同期」という用語は、１つのスレッドが、オブジェクト、変数、またはデータに対する動作を中断されることなく確実に完了することができるようにする能力を指す。セマフォまたはｍｕｔｅｘｅｓを使用して、同期を実施することができる。別法として、スレッド・アクセスの間の同期は、関連するモニタによって保護されたコード領域を一度にただ１つのスレッドのみが実行することができるようにする高水準機構である、「モニタ」を使用することによって得ることもできる。Ｊａｖａ™ 言語では、スレッドをブロックし、その後それが利用可能になったときにブロックしたスレッドに再度通知することができるオブジェクトは、モニタ・オブジェクトと呼ばれる。例えば第１図を参照すると、オブジェクト

１６がモニタ・オブジェクトになることもあり、その場合には、１４ａがオブジェクト１６へアクセスするときには、スレッド１４ｂ～ｃをブロックし、スレッド１４ａがオブジェクト１６へのアクセスを完了したときにスレッド待ち行列１２に通知することになる。その他のスレッドによるオブジェクトへのアクセスをブロックする処置は、「ロック」を実行すると呼ばれ、別のスレッドによるオブジェクトへのアクセスを再度可能にする処置は、「ロック解除」動作と呼ばれる。

同期メソッドを実行するスレッドには、常に単一のスレッドしかオブジェクトにアクセスすることができず、したがって特定の並行操作が禁止されることがあるという性能の不利があることは理解されるであろう。

オブジェクト１６内に組み込まれた共通データ構造タイプは第２図に示すリンク済みリストのデータ構造２０である。このリンク済みリストのデータ構造２０は、順序づけられたエレメント２２ａ～２２ｎのシーケンスを含み、このシーケンスへのエレメントの迅速な挿入、およびそこからのエレメントの迅速な除去を容易にする。各エレメント２２は、データ項目２４、およびリンク済みリストのデータ構造２０中の次の順次エレメント２２へのポインタ２６を含む。データ構造２０中の最後のエレメント２２ｎは、リストの終了を表すヌル・ポインタを有する。

第３ａ図および第３ｂ図は、リンク済みリストのデータ構造２０の修正形態を示す図である。具体的に言うと、第３ａ図は、データ構造２０中のエレメント２２（ｎ）と２２（ｎ＋２）の間へのエレメント２２（ｎ＋１）の挿入を示す図である。第３ｂ図は、同様に、リンク済みリストのデータ構造２０からのエレメント２２（ｎ＋１）の除去を示す図である。まず第３ａ図を参照すると、リンク済みリストのデータ構造２０を含むオブジェクトへのアクセスを有する活動状態のスレッドは、エレメント２２（ｎ＋１）を挿入しようとすることができる。このようなエレメントをリンク済みリストのデータ構造２０に挿入（または追加）するには、エレメント２２（ｎ＋１）を挿入しようとする位置の直前にあるエレメント２２（ｎ）のポインタを、以前の順次エレメント２２（ｎ＋２）から新しく

挿入されたエレメント２２（ｎ＋１）を指すように更新する必要がある。このエレメント２２（ｎ）のポインタの更新を、矢印２８で示す。挿入されたエレメント２２（ｎ＋１）のポインタも、エレメント２２（ｎ＋２）を指すようにセットしなければならない。

本発明の１実施形態では、各ポインタ２６は３２ビット・アドレスを含む。プログラム中での同期の必要は、２つのスレッドが同時にリンク済みリストのデータ構造２０にアクセスしているという仮定を考慮すると明らかとなる。具体的に言うと、エレメント２２（ｎ＋１）を挿入している第１スレッドが、エレメント２２（ｎ）のポインタ２６を更新し、新しく挿入されたエレメント２２（ｎ＋１）を指すようにしている場合を考慮する。これと同時に、第２スレッドが、それによって指定されたデータ・パスを検査することによってデータ構造２０のトラバースを実行していることもある。具体的に言うと、第２スレッドが、所定のデータ項目２４を検索するためにデータ構造２０をトラバースしていることもある。第２スレッドは各エレメントのポインタに依存してそれをシーケンス中の次のエレメントに向けるので、部分的に修正されたポインタ上で実行される読取り動作は、第２スレッドによるデータ構造２０のトラバースに不利であることが分かることは理解されるであろう。上記に概説したシナリオを防止するために、リンク済みリストのデータ構造２０にアクセスするオブジェクトのメソッドを同期させ、ただ１つのスレッドまたはメソッドのみが常にデータ構造２０へのアクセスを有するようにすることが一般に行われている。具体的に言うと、同期を利用して、リンク済みリストのデータ構造を参照するハッシュ・テーブルなどのデータ構造上で、またはリンク済みリストのデータ構造２０上で直接、ロックを達成することができる。ただし、一度にただ１つのメソッドしかリンク済みリストのデータ構造にアクセスすることができないようにすることにより、最適な性能が提供されなくなることは理解されるであろう。本発明の意義が理解されるようにするには、リンク済みリストのデータ構造２０に対して実行することができる下記の３つの動作について考慮しなければならない。すなわち、

１．第３ａ図に示すようにエレメントがリンク済みリストのデータ構造２０に追加される、挿入（または追加）動作、

２．第３ｂ図に示すようにエレメントがデータ構造２０から除去される、除去（または削除）動作、および

３．ある特定のデータ項目がその中に存在することを確定する、かつ／または場合によってはこのようなデータ項目を検索するためにデータ構造２０内のエレメントの内容が検査される、トラバース動作。

トラバース動作は、データ構造２０への修正が起こらないという点で、挿入動作および除去動作とは区別される。本発明は、リンク済みリストのデータ構造２０の非修正トラバースがその他のトラバース動作および修正動作と非同期である方法を提案する。その非同期トラバース動作を実施すると、修正動作とトラバース動作が同時に起こることができ、またリンク済みリストのデータ構造２０の非同期トラバースもいくつか同時に起こることができるので有利である。したがって、複数のスレッドがリンク済みリストのデータ構造２０について主としてトラバース動作を実行するプログラムでは、これらの多数のスレッドが同時にデータ構造２０にアクセスすることができる。非同期トラバース動作の実施は、エレメント２２のポインタ２６のスレッドによるアトミックな読取りおよび書込み動作を指定することによって容易になる。具体的に言うと、アトミック動作は機能的に不可分であり、したがって確実に完了する。次に本発明の方法論について、第３ａ図および第３ｂ図を特に参照して記述する。第３ａ図に示す挿入メソッドは（その他の同期メソッドと）同期しており、特に下記の段階を含む。

１．挿入しようとするエレメントが既にリンク済みリスト構造２０内に存在するかどうかを検査する段階。存在する場合には挿入動作を打ち切る。

２．アトミック動作において、挿入されるエレメント２２（ｎ＋１）のポインタを更新して次の順次エレメント２２（ｎ＋２）を指すようにする段階。この段階が完了すると、エレメント２２（ｎ）および２２（ｎ＋１）はともに同じエレメント２２（ｎ＋２）を指している。

３．アトミック動作において、エレメント２２（ｎ）のポインタを更新し、挿入されたエレメント２２（ｎ＋１）を指すようにする段階。

同様に、リンク済みリストのデータ構造２０からエレメント２２を除去する同期メソッドは、特に下記の段階を含むことができる。

１．検査動作を実行して、関連するデータ項目が実際にリンク済みリスト中に含まれるかどうかを確定する段階。

２．アトミック動作において、エレメント２２（ｎ）のポインタを、エレメント２２（ｎ＋２）を指すようにセットする段階。

上記で実行した動作において、エレメント２２のポインタ２６の修正がアトミック動作を含むことに留意されたい。したがって、こうした挿入または削除動作の間にリンク済みリストのデータ構造２０のトラバース動作を実行するいかなるメソッドも、有効なデータ・パスを検査することが保証されることになる。非同期トラバース動作を実施するメソッドによるリンク済みリストのデータ構造２０へのアクセスを容易にするのは、この有効データ・パスの保証である。ただし、１実施形態では、このような修正動作間の衝突を防止するために同期修正動作が依然として必要であることは理解されるであろう。したがって、非同期トラバース・メソッドを使用することで、このような非同期トラバース・メソッドを組み込むいくつかのスレッドが単一のリンク済みリストのデータ構造２０に同時にアクセスすることが可能となり、こうしてコンピュータ・プログラムの性能を最適化することが可能となる。

さらに本発明は、いわゆる「マーク／スイープ・ガーベージ・コレクション」を実施する能力を有するプログラミング言語での使用に特に適している。具体的に言うと、このようなマーク／スイープ・ガーベージ・コレクション技術では、特定のオブジェクトへの参照のトラッキングを維持するために参照カウンタ機構を実装する必要がない。参照カウントを利用するガーベージ・コレクション技法は同期動作を必要とし、したがって同期ガーベージ・コレクション動作を実行するときに、本発明が達成する性能上の利点を制限することがある。

次に第４図を参照すると、本発明によるいくつかのエレメント２２を含むリンク済みリストのデータ構造２０を管理する方法３０が示されている。この方法は段階３２から開始し、進行して２組の動作を並列に実行する。具体的に言うと、方法３０では、段階３４で第１スレッドはエレメント２２を挿入または除去することによってリンク済みリストのデータ構造２０を修正することができる。次いで方法３０は段階３６に進み、データ構造２０内の関連するエレメント２２のポ

インタをアトミックに修正し、段階３４で行われた修正を反映する。本発明の１実施形態では、ポインタ２６のアトミックな修正および読取りを容易にするプログラミング言語は、Ｊａｖａ™ プログラミング言語である。段階３４および３６を実行するのと同時に、本発明の方法３０では、段階３８で、リンク済みリストのデータ構造２０の任意数の非同期トラバース動作を並行して実行することができる。

第５図は、それぞれのリンク済みリストのデータ構造２０をそれぞれ参照することができるいくつかのエントリまたはバケット４２を含む、ハッシュ・テーブル４０を示す図である。ハッシュ・テーブル・データ構造では、オブジェクトとなるデータ項目（例えば従業員記録）はキーで識別される。例えば、リンク済みリストのデータ構造２０ａの第１エレメント２２中のデータ項目は、キー「ＡＡＡ」で識別される。ハッシュ・テーブル４０は、ハッシュ・コードと呼ばれる、キーから得られる整数値を計算する。したがって、図示の例のキー「ＡＡＡ」から生成されたハッシュ・コードはゼロ（０）となる。こうしてハッシュ・テーブル４０は、短縮した配列指標の生成を容易にする。共通したハッシュ・コードが生成されるキーを有するデータ項目は同一のバケットに入っているものと考えられ、第５図に示すものなどのリンク済みリストのデータ構造２０を含むことができる。本発明は、ハッシュ・テーブル４０によって参照されるリンク済みリストのデータ構造２０を管理するために利用することができる。

第６図は、本発明による、クラス５２のオブジェクトを含む一連のオブジェクト５０を示す図である。各オブジェクト５０は、いくつかのリンク済みリストのデータ構造２０に指標を付けるハッシュ・テーブル構造４０を含むものとして示してある。各オブジェクトは、３つの非同期トラバース・メソッド、すなわち「ＣＯＮＴＡＩＮＳ」メソッド６０、「ＣＯＮＴＡＩＮＳ　ＫＥＹ」メソッド６２、および「ＧＥＴ」メソッド６４をさらに含む。「ＣＯＮＴＡＩＮＳ」メソッド６０は、スレッドによって与えられたキーが、ハッシュ・テーブル・データ構造４０中の指定された値にマップされるかどうかをテストする。「ＣＯＮＴＡＩＮＳ　ＫＥＹ」メソッド６２は、指定されたオブジェクトがハッシュ・テーブル・データ構造４０中のキーであるかどうかをテストし、「ＧＥＴ」メソッド６４

は、与えられたキーがハッシュ・テーブル・データ構造４０中でマップされるデータ項目を戻す。各オブジェクトは、上述の方法論に従ってリンク済みリストのデータ構造２０中にエレメント２２を挿入することができる、同期「ＰＵＴ」メソッド６６をさらに含む。各オブジェクト５０は、上述の方法論に従ってリンク済みリストのデータ構造２０からエレメント２２を除去することができる、同期「ＲＥＭＯＶＥ」メソッド６８をさらに含む。もちろん、上述のオブジェクト５０は単なる例示であり、本発明の原理は様々な構造の任意数のオブジェクトに適用することができることは理解されるであろう。

次に第７図には、プロセッサ７２、スタティック・メモリ７４、およびメイン・メモリ７６を含むコンピュータ・システム７０が示してある。プロセッサ７２ならびにメモリ７４および７６は、バス７８を介して、互いに、またいくつかの周辺機器と連絡している。上述の周辺機器には、ビデオ・ディスプレイ８０（陰極線管（ＣＴＲ）や液晶ディスプレイ（ＬＣＤ）など）、英数字入力装置８２（キーボードなど）、カーソル制御装置８４（マウスなど）、コンピュータ可読媒体８８を収容する駆動ユニット８６、信号発生装置９０（１対の音声スピーカなど）、およびネットワーク・インタフェース装置９４を含む。本発明では、「コンピュータ可読媒体」という用語は、駆動ユニット８６、メイン・メモリ７６、スタティック・メモリ７４、プロセッサ７２、またはプロセッサ７２が実行するためのデータを得ることができるその他任意の媒体の中に収容することができる、磁気記憶装置を指すものとする。上述のものなどのメソッドを含むハッシュ・テーブル・オブジェクト９２は、コンピュータ可読媒体８８、メイン・メモリ７６、またはプロセッサ７２自体の中に完全または部分的に常駐するものとして示されている。ハッシュ・テーブル・オブジェクト９２は、プロセッサ７２によって実行されたときに少なくとも第４図に関連して記述した段階をプロセッサ７２に実行させる、一連の命令を含む。

ネットワーク・インタフェース装置９４は、モデム、ネットワーク・アダプタ・カード、またはコンピュータ・システム７０をコンピュータ・ネットワークに結合するその他任意の装置にすることができる。ネットワーク・アクセス装置９４を使用して、コンピュータ・データ信号がコード化された搬送波を発生または

受信することができる。コンピュータ・データ信号を解釈して、本発明を実施するために実行することができるプログラム・コードを生成することができる。

このように、リンク済みリストのデータ構造を管理する方法について記述した。特定の例示的な実施形態に関連して本発明を説明したが、本発明の広範な主旨および範囲を逸脱することなくこれらの実施形態に様々な修正および変更を加えることができることは明らかであろう。したがって、本明細書および図面は、制限的なものではなく例示的なものと見なされたい。

【図１】

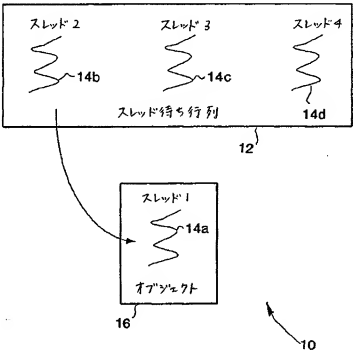

FIG. 1

【図2】

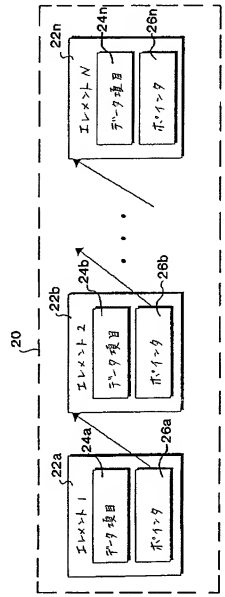
20
22a
エレメント 1
データ項目
ポインタ
24a
26a
22b
エレメント 2
データ項目
ポインタ
24b
26b
22n
エレメント N
データ項目
ポインタ
24n
26n

FIG. 2

【図3】

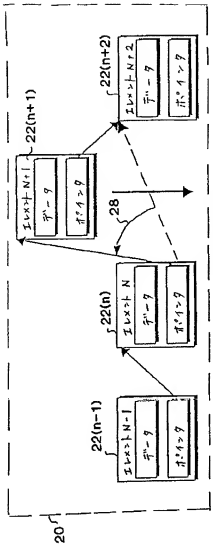

FIG. 3a

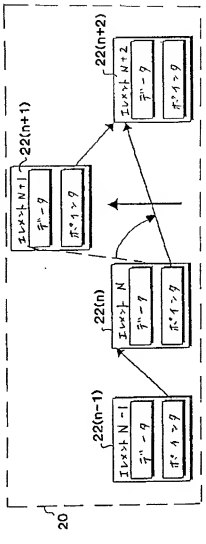

FIG. 3b

【図４】

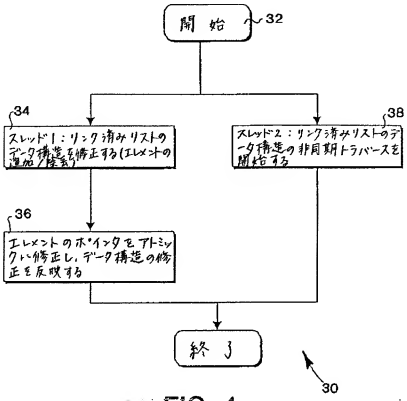
開始
32
34
スレッド1：リンク済みリストのデータ構造を修正する（エレメントの追加／除去）
38
スレッド2：リンク済みリストのデータ構造の非同期トラバースを開始する
36
エレメントのポインタをアトミックに修正し，データ構造の修正を反映する
終了
30

FIG. 4

【図5】

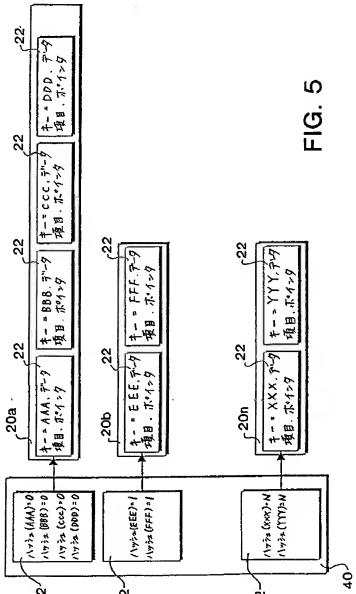
20a
20b
20n
22
42
40
キー＝AAA，データ
項目，ポインタ
キー＝BBB，データ
項目，ポインタ
キー＝CCC，データ
項目，ポインタ
キー＝DDD，データ
項目，ポインタ
キー＝EEE，データ
項目，ポインタ
キー＝FFF，データ
項目，ポインタ
キー＝XXX，データ
項目，ポインタ
キー＝YYY，データ
項目，ポインタ

FIG. 5

【図6】

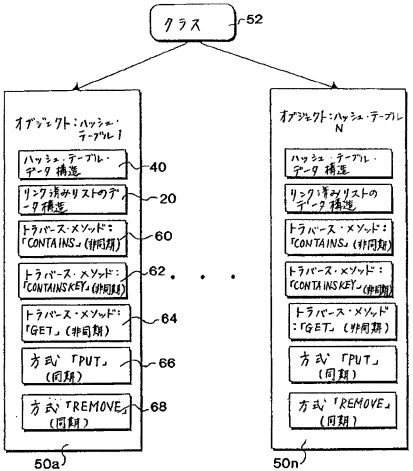
クラス
52
オブジェクト：ハッシュ・テーブル1
ハッシュ・テーブル・データ構造
40
リンク済みリストのデータ構造
20
トラバース・メソッド：「CONTAINS」(非同期)
60
トラバース・メソッド：「CONTAINSKEY」(非同期)
62
トラバース・メソッド：「GET」(非同期)
64
方式「PUT」(同期)
66
方式「REMOVE」(同期)
68
50a
オブジェクト：ハッシュ・テーブルN
ハッシュ・テーブル・データ構造
リンク済みリストのデータ構造
トラバース・メソッド：「CONTAINS」(非同期)
トラバース・メソッド：「CONTAINSKEY」(非同期)
トラバース・メソッド：「GET」(非同期)
方式「PUT」(同期)
方式「REMOVE」(同期)
50n

FIG. 6

【図7】

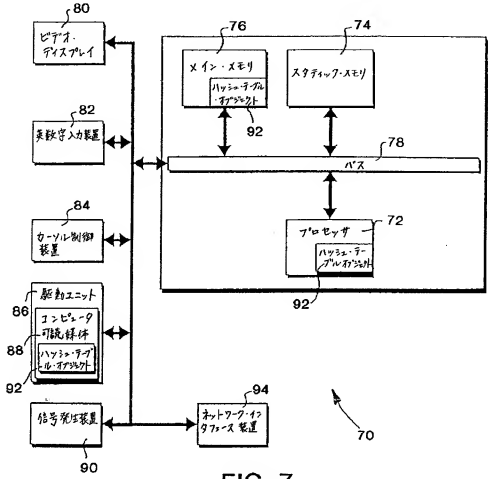
80
ビデオ・ディスプレイ
82
英数字入力装置
84
カーソル制御装置
86
88
92
駆動ユニット
コンピュータ可読媒体
ハッシュ・テーブル・オブジェクト
信号発生装置
90
76
メイン・メモリ
ハッシュ・テーブル・オブジェクト
74
スタティック・メモリ
92
78
バス
72
プロセッサ
ハッシュ・テーブル・オブジェクト
92
94
ネットワーク・インタフェース装置
70

FIG. 7

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/13652

A. CLASSIFICATION OF SUBJECT MATTER
IPC(6) :G06F 9/18, 15/40, 13/00, 12/14, 17/30, 13/14.
US CL :707/8, 100, 203, 206
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED
Minimum documentation searched (classification system followed by classification symbols)
U.S. : 707/8, 100, 203, 206

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
APS, IEEE

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 4,149,243 A (WALLIS) 10 April 1979, col. 2, lines 1-52.	1-24
Y	US 5,168,554 A (LUKE) 01 December 1992, col. 2, lines 31-32.	1-24
Y	US 5,442,758 A (Slingwine et al) 15 August 1995, col. 5, lines 25-67.	1-24
Y	US 5,293,600 A (VRIEZEN) 08 March 1994, col. 3, line 30-col. 4, line 47].	1-24.
Y	US 5,469,567 A (OKADA) 21 November 1995, col. 3, lines 10-43.	1-24.
Y	US 5,495,609 A (SCOTT) 27 February 1996, col. 2, line 35-col. 3, line 35.	1-24

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 18 SEPTEMBER 1998	Date of mailing of the international search report 29 OCT 1998
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer FRANTZ COBY Telephone No. (703) 305-9707

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/13652

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,287,521 A (NITTA et al) 15 Febraury 1994, col. 3, lines 40-67.	1-24
Y	US 5,295,262 A (SEIGH, II) 15 March 1994, col. 2, lines 17-29.	1-24
Y	US 5,319,778 A (CATINO) 07 June 1994, col. 3, line 5-49).	1-24

【要約の続き】

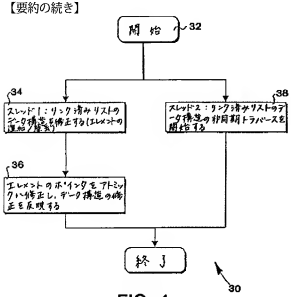
開始
32
34
スレッド1：リンク済みリストのシンク構造を修正する（エレメントの追加/削除）
38
スレッド2：ランク済みリストのデータ構造の非同期トラバースを開始する
36
エレメントのポインタをアトミックに修正し、データ構造の修正を反映する
終了
30

FIG. 4